(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Offenlegungsschrift**

(10) **DE 100 06 416 A 1**

(51) Int. Cl.[7]: G 06 F 13/10

(21) Aktenzeichen: 100 06 416.7
(22) Anmeldetag: 14. 2. 2000
(43) Offenlegungstag: 2. 11. 2000

DE 100 06 416 A 1

(30) Unionspriorität:
298590 23. 04. 1999 US

(71) Anmelder:
Hewlett-Packard Co., Palo Alto, Calif., US

(74) Vertreter:
Schoppe, Zimmermann & Stöckeler, 81479 München

(72) Erfinder:
Simpson, Shell S., Boise, Id., US; Clough, James, Meridian, Id., US; Gartner, M. Scott, Boise, Id., US

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) System und Verfahren für eine Peripheriesystemverwaltung unter Verwendung von Mehr-Objekt-Schnittstellen

(57) Die Erfindung ermöglicht, daß ein Computer Elemente eines Peripheriesystems verwaltet, wobei jede Peripherieeinheit des Systems einen oder mehrere Dienste für den Computer liefert. Jede Peripherieeinheit weist eine zugeordnete "Verwaltete-Entität"-Schnittstelle (ME-Schnittstelle) für jeden Dienst auf. Jede ME-Schnittstelle umfaßt einen Bezug (Bezüge) auf eine oder mehrere "Verwaltungsschnittstellen" (MI), die aus mehreren Verfahren zum Liefern von Informationen bezüglich eines Merkmals (von Merkmalen) der ME und/oder zum Bewirken, daß das Gerät, das die ME aufweist, eine Funktion durchführt, besteht. Der Computer umfaßt ein Eingabe/Ausgabe-Modul zum Ermöglichen von Kommunikationen mit jeder der Einheiten des Peripheriesystems und einen Speicher zum Speichern eines Explorer-Objekts. Ein Prozessor in dem Computer spricht auf eine Benutzerabfrage bezüglich einer Peripherieeinheit an, um (i) das Explorer-Objekt aufzurufen, um eine ME zu bestimmen, die der Peripherieeinheit entspricht, und (ii) ein Verfahren des Schnittstellenanbieter-Objekts aufzurufen, um ein MI-Objekt wiederzugewinnen, das der ME zugeordnet ist, das Informationen und Verfahren, die sich auf die Peripherieeinheit, die Aktivitäten derselben und die Benutzerabfrage beziehen, umfaßt.

DE 100 06 416 A 1

Beschreibung

Diese Erfindung bezieht sich auf die Verwaltung von Peripheriesystemen und spezieller auf ein System und ein Verfahren für eine Peripheriesystemverwaltung, die Objektschnittstellen verwendet.

Ein Verwalten eines Peripheriesystems beinhaltet ein Konfigurieren von Geräten, aus denen das Peripheriesystem besteht, ein Erhalten des Zustands derselben und ein Empfangen von Ereignissen von den Geräten. Die Fähigkeiten der Geräte variieren stark und Kommunikationsprotokolle, die verwendet werden, um mit den Geräten zusammenzuwirken, variieren ebenfalls stark. Eine Software, die ein Peripheriesystem verwaltet, muß diese Vielfalt unterbringen und ist folglich oft ziemlich komplex. Ein Peripheriesystem, wie der Ausdruck später verwendet wird, umfaßt Drucker, Scanner (Abtastvorrichtungen), Server, Multifunktionsgeräte und Peripherie-Clients, wie beispielsweise Personalcomputer.

Verkäufer liefern dauernd neue Modelle von Peripheriegeräten, Peripherieservern und Peripherie-Clients die weitere Herausforderungen an Entwickler von Peripheriesystemverwaltungs-Softwaresystemen darstellen. Eine solche Software muß dauernd überprüft werden, um solche neuen Geräte und neuen Gerätemerkmale zu unterstützen, was ferner die Komplexität solcher Software erhöht.

Die Fähigkeiten, die durch Peripheriegeräte, Peripherieserver und Peripherie-Clients geliefert werden, variieren basierend auf deren jeweiligen Konfigurationen. Wenn Betriebsmittel, wie beispielsweise Festplatten, hinzugefügt oder entfernt werden, muß die Peripheriesystemverwaltung diese Fähigkeitsänderungen handhaben – was ferner die Komplexität der Peripheriesystemverwaltungssoftware erhöht.

Peripheriesystemverwaltungs-Anwendungen (PSM-Anwendungen; PSM = peripheral system management) ermöglichen oft Endbenutzern, "Operationen" bezüglich einer Ansammlung von Geräten unter Verwendung eines einzigen Befehls durchzuführen, was im Gegensatz zum Ausgeben des selben Befehls immer wieder für jedes Gerät steht. Zum Beispiel ermöglichen PSM-Anwendungen, daß einer Ansammlung von Geräten ihre Adressen durch eine Verwendung eines einzigen Befehls neu zugewiesen werden, wobei jedoch solche Anwendungen keine neuen Fähigkeiten bezüglich solcher Geräte unterbringen. PSM-Anwendungen zeigen ferner typischerweise Benutzerschnittstellenelemente an, die eine Ansammlung von Geräten, die verwaltet werden, darstellen. Durch diese Schnittstelle kann der Benutzer Verwaltungshandlungen bezüglich mehrerer Geräte zur gleichen Zeit durchführen. Im allgemeinen ist diese Schnittstelle nicht erweiterbar und liefert nicht die Fähigkeit; die Art, auf die eine PSM-Anwendung mehrere Geräte anzeigt, signifikant zu ändern. Im Gegensatz ermöglichen PSM-Anwendungen, daß Geräteentwickler "Applet"-Erweiterungen (d. h. kleine Programme) beitragen, die spezifizieren, wie ein Gerät unterstützt wird. Wenn ein Benutzer einen Wunsch anzeigt, ein einzelnes Gerät zu verwalten, wird das Applet verwendet, um eine gerätespezifische Benutzerschnittstelle für den Benutzer darzustellen. Auf dieselbe Art, auf die ein Geräteentwickler die Art erweitern kann, auf die ein einzelnes Gerät betrachtet wird, existiert ein Bedarf danach, eine Änderung der Art, auf die eine Ansammlung der Geräte dargestellt wird, zu ermöglichen.

Demgemäß existiert ein Bedarf nach einem Peripheriesystemverwaltungssystem, das fähig ist, mit mehreren Typen von Geräten in dem Peripheriesystem zusammenzuwirken; das auf Gruppen von Geräten wirken kann, die dieselben Fähigkeiten liefern; das fähig ist, zu bestimmen, ob ein Element des Systems eine besondere Fähigkeit liefert oder nicht; das erweiterbar ist, um neu eingebaute Gerätefähigkeiten unterzubringen, sowohl vor als auch nach einem Verkauf an einen Kunden; das Änderungen der Art und Weise unterbringen kann, auf die Gruppen von Geräten dargestellt werden; und das ausreichend dynamisch ist, um zu ermöglichen, daß die Fähigkeiten der Peripheriesystemelemente während des Betriebs auftreten und verschwinden.

Die Aufgabe der vorliegenden Erfindung besteht darin, ein System und ein Speichermedium zu schaffen, die eine flexible Verwaltung einer Mehrzahl von Einheiten eines Peripheriesystems ermöglichen.

Diese Aufgabe wird durch ein System gemäß Anspruch 1 und ein Speichermedium gemäß Anspruch 9 gelöst.

Die Erfindung ermöglicht, daß ein Computer Elemente eines Peripheriesystems verwaltet, wobei jede Peripherieeinheit einen oder mehrere Dienste für den Computer liefert. Jede Peripherieeinheit weist eine zugeordnete "Verwaltete-Entität"-Datenstruktur (ME-Datenstruktur; ME = Managed Entity) für jeden Dienst auf. Jede ME-Datenstruktur umfaßt einen Bezug (Bezüge) auf ein oder mehrere "Verwaltungsschnittstellen"-Objekte (MI-Objekte; MI = Management Interface), die aus einem oder mehreren Verfahren zum Liefern von Informationen bezüglich eines Merkmals (Merkmalen) der ME und/oder zum Bewirken, daß das Gerät, das die ME aufweist, eine Funktion durchführt, bestehen. Der Computer umfaßt ein Eingabe/Ausgabe-Modul zum Ermöglichen von Kommunikationen mit jeder der Einheiten des Peripheriesystems und einen Speicher zum Speichern eines "Explorer"-Objekts (Erforscher-Objekt). Ein Prozessor in dem Computer spricht auf eine Benutzerabfrage bezüglich einer Peripherieeinheit an, um (i) das Explorer-Objekt aufzurufen, um eine ME, die der Peripherieeinheit zugeordnet ist, zu bestimmen, und (ii) ein Verfahren eines Schnittstellenanbieterobjekts aufzurufen, um ein MI-Objekt, das der ME zugeordnet ist und Informationen und Verfahren, die sich auf die Peripherieeinheit, die Aktivitäten derselben und die Abfrage des Benutzers beziehen, umfaßt, wiederzugewinnen.

Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung werden nachfolgend unter Bezugnahme auf die beigefügten Zeichnungen näher erläutert. Es zeigen:

Fig. 1 eine schematische Darstellung einer typischen ME-Datenstruktur;

Fig. 2 eine vereinheitlichte Modellierungsprachen-Darstellung der Beziehungen zwischen einer ME-Klasse und einer MI-Klasse, die zeigt, daß die ME eine Unterklasse der MI ist;

Fig. 3 unterschiedliche Arten von MEs;

Fig. 4 ein Beispiel der Ableitung von ME-Klassen und ferner zwei MIs, die nicht auch MEs sind. **Fig.** 4 zeigt ferner, wie eine vielfache Vererbung durch eine ME verwendet werden kann;

Fig. 5 ein Blockdiagramm eines Systems zum Durchführen des Verfahrens der Erfindung; und

Fig. 6 und **Fig.** 7 ein logisches Flußdiagramm eines Beispiels des Verfahrens der Erfindung.

Eine Netzsoftware muß mit physischen Geräten zusammenwirken, die mit dem Netz verbunden sind. In der Vergangenheit wurde dies durch ein Beschreiben einer physischen Topologie und ein Zuweisen eines Namens zu jeder Maschine erreicht. Jedoch kann eine Software, die an vielen Orten entwickelt wird, auf diese Weise nicht beschrieben wer-

den, da physische Topologien von Ort zu Ort variieren. Diese Erfindung verwendet eine logische Topologie, die die Rollen (oder Dienste) beschreibt, die Maschinen spielen/liefern, und spezifiziert nicht spezielle Maschinen.

Beispiele von Rollen, die Maschinen in einer Peripheriesystemverwaltungs-Software (PSM-Software) spielen, die die Erfindung verkörpert, sind im folgenden angegeben:

- "Verwaltungs-Client-Maschine": Die Rolle einer Maschine, die eine Benutzerschnittstelle für eine PSM-Software darstellt. Zum Beispiel wird die Rolle des Verwaltungs-Client durch die Maschine durchgeführt, die ein Netzsuchprogramm (Web-Browser) ausführt, das verwendet wird, um auf ein Netzadministrationsprogramm zuzugreifen.
- "Verwaltungsservermaschine": Die Rolle einer Maschine, die eine Logik für eine PSM-Software liefert. Zum Beispiel führt die Servermaschine, in der ein Netzadministrationsprogramm installiert ist, die Rolle einer Verwaltungsservermaschine aus.
- "Peripherie-Client-Maschine": Die Rolle einer Maschine, die Dienste verwendet, die durch ein Peripheriegerät geliefert werden. Eine Endbenutzerworkstation, die einen Drucker verwendet, ist ein Beispiel einer Peripherie-Client-Maschine.
- "Peripherieservermaschine": Die Rolle einer Maschine, die einen Zugriff auf die Dienste, die durch ein Peripheriegerät geliefert werden, vermittelt. Eine Windows-NT-Maschine (eine Marke der Microsoft Corporation), die ihre Drucker teilt, ist ein Beispiel einer Peripherieservermaschine.
- "Peripheriemaschine": Die Rolle einer Maschine, die als ein Peripheriegerät dient. Drucker und Scanner sind Beispiele von Peripheriegeräten.
- "Peripherie-Verzeichnis-Maschine": Die Rolle einer Maschine, die einen Dienst liefert, der verwendet wird, um Peripheriemaschinen und Peripherieservermaschinen zu finden.

Es sei angemerkt, daß eine physische Maschine mehrere Rollen durchführen kann und mehrere Dienste liefern kann. Zum Beispiel kann ein zentraler Server sowohl als eine Verwaltungsservermaschine (eine Maschine, in der ein Netzadministrationsprogramm installiert ist) als auch als eine Peripherieservermaschine (z. B. ein Druckserver) dienen.

Die Peripherieverwaltungssoftware, die die Erfindung beinhaltet, ist fähig, MEs, die sich auf Peripheriegeräte beziehen, zu entdecken, zu organisieren, anzuzeigen, zu modifizieren und zu überwachen. Eine ME ist ein Softwareobjekt, durch das die Rolle eines Geräts und ein Dienst, den das Gerät liefert, verwaltet werden können. Es sei erneut betont, daß ein einzelnes Gerät mehrere Rollen haben kann und somit mehreren MEs zugeordnet sein kann.

- Eine ME ist eine Architektur-Abstraktion, die verwendet wird, um etwas zu modellieren, das verwaltet wird.
- MEs werden durch eine "Explorer"-Architektur-Abstraktion entdeckt.
- MEs werden durch eine "Datenbank"-Architektur-Abstraktion organisiert.
- MEs werden durch eine "Ansicht"-Architektur-Abstraktion angezeigt.
- MEs werden durch eine "Operation"-Architektur-Abstraktion modifiziert.
- MEs werden unter Verwendung von "Ereignis"- und "Handlung"-Architektur-Abstraktionen überwacht.

Wie aus der folgenden Erörterung klar wird, ist jede der Abstraktionen, die oben identifiziert sind, als ein Softwareobjekt konfiguriert, das in dem klassischen Sinn seinen zustand in "Variablen" beibehält und sein Verhalten mit einem oder mehreren Verfahren implementiert. Die verschiedenen Objekte, die unten beschrieben werden, versehen die PSM-Software, die die Erfindung beinhaltet, mit einer beträchtlichen Flexibilität, um System- und Software-Modifikationen unterzubringen.

Verwaltete Entitäten

Definition

Eine ME (verwaltete Entität) ist ein Objekt, das ein benutzererkennbares Element in der Peripherieverwaltungsproblemdomäne darstellt. Eine ME liefert eine Ansammlung von einer oder mehreren Verwaltungsschnittstellen (MIs). Eine MI ist ein Satz von einer oder mehreren Funktionen, die verwendet werden, um mit einer ME zusammenzuwirken (oder diese zu identifizieren).

Ein Benutzer, der für das Verwalten von Druckern verantwortlich ist, erkennt die Existenz von Druckern, Druckservern und Druck-Clients. Mit oder ohne PSM-Software wird dieser Benutzer mit jedem dieser Elemente interagieren, um Probleme zu lösen. Derselbe Benutzer wird wahrscheinlich die detaillierte Architektur eines Spoolers nicht kennen. Die Existenz einer Torüberwachungsvorrichtung oder eines Druckprozessors wird aller Wahrscheinlichkeit nach für einen solchen Benutzer irrelevant sein. Bei dem Lösen von Problemen setzt sich dieser Benutzer lediglich mit denjenigen Problemen auseinander, die ihm vertraut sind. Diese vertrauten und einfach erkannten Elemente sind die Basiseinheit der Verwaltung, von der Benutzer erwarten, daß sie dieselbe in der PSM-Software finden. Die Objekte, die diese Basiseinheiten der Verwaltung darstellen, werden MEs genannt.

Jede dieser Basiseinheiten der Verwaltung wird in zusätzliche Elemente unterteilt. Zum Beispiel ist ein Druckserver aus einem Spooler, der aus noch weiteren Elementen gebildet ist, wie Treibern, Torüberwachungseinrichtungen und Druckprozessoren, gebildet. Obwohl es möglich ist, daß jedes dieser Elemente verwaltet werden muß, werden diese typischerweise im Zusammenhang mit dem Verwalten des Druckservers verwaltet. Ein Benutzer rüstet den Treiber auf einem besonderen Druckserver durch eine Interaktion mit dem Druckserver nach.

Die Ansammlung von MIs, die durch eine ME geliefert wird, ist erweiterbar. Zum Beispiel kann ein Drucker eine Schnittstelle zum Verwalten von Merkmalen des Druckers (d. h. Hinzufügen, Entfernen) liefern, sogar wenn diese Schnittstelle vorher nicht in der PSM-Software geplant war. Um eine Schnittstelle zu verwenden, sind "Operation"- und

"Ansicht"-Objekte (die unten beschrieben werden) vorgesehen.

Die Ansammlung von MIs, die durch eine ME geliefert wird, ist ferner dynamisch. Während des Betriebs kann die Ansammlung von MIs modifiziert werden. Als Betriebsmittel, die für eine ME-Änderung verfügbar sind, können die MIs, die durch diese ME geliefert werden, geändert werden. Zum Beispiel kann ein Drucker eine Festplatte aufweisen oder nicht. Da die Anwesenheit einer Festplatte bestimmt, ob es möglich ist, Massenspeicherschriftarten zu verwalten oder nicht, wird die MI, die sich auf eine Verwaltung von Massenspeicherschriftarten bezieht, nicht geliefert, falls keine Festplatte verfügbar ist.

MIs (und die Funktionen, die dieselben aufweisen), die durch eine besondere ME unterstützt werden, können während des Betriebs abgefragt werden. Eine Anwendung, die die MIs abfrägt, muß nicht notwendigerweise wissen, wie dieselben zu verwenden sind. Die Anwendung kann statt dessen einfach verfolgen, welche MIs jede ME liefert, und diese Informationen verwenden, um MIs mit Komponenten, die diese Schnittstellen verstehen, auf Übereinstimmung prüfen. Zum Beispiel kann eine Anwendung, die nichts über die MI, die verwendet wird, um Schriftarten zu verwalten, weiß, eine "Operation", die die Schriftartverwaltungsschnittstelle erfordert, mit MEs, die diese Schnittstelle liefern, auf Übereinstimmung prüfen.

Verwaltungsschnittstellen

Normalerweise weist ein Objekt eine einzelne, statische Schnittstelle auf. Jedoch bedeutet das Aufweisen lediglich einer Schnittstelle, daß die Schnittstelle jedesmal überarbeitet werden muß, wenn die Fähigkeiten des Objekts geändert werden. Vielfache Überarbeitungen derselben Schnittstelle führen zu Konfusionen. Dies wird gemäß dieser Erfindung durch ein Sicherstellen, daß eine ME mit mehreren Schnittstellen durch ein oder mehrere Objekte dargestellt wird, vermieden, von denen jedes als "Verwaltungsschnittstelle" (MI) bezeichnet wird. Eine MI weist ein oder mehrere Verfahren zum Bilden von Schnittstellen mit einer spezifischen Fähigkeit der ME auf. Demgemäß können andere einzelne Objekte diese MIs wiederverwenden. Ferner können diese MIs während des Betriebs – ohne ein Neukompilieren – auftreten und verschwinden.

Die Granularität von MIs ist eine Entwurfsentscheidung, die nicht ihre Nutzbarkeit beeinflußt. Eine MI kann einen großen Satz von voneinander unabhängigen Funktionen oder einen kleinen Satz von eng verwandten Funktionen umfassen. Zum Beispiel kann eine MI, die "Neuer-LaserJet" genannt wird, definiert sein, die Funktionen für Merkmale, die in allen neuen LaserJet-Druckern (LaserJet ist eine Marke der Hewlett-Packard Company) gefunden werden, umfaßt. Derselbe Satz von Funktionen kann ferner in mehreren MIs plaziert werden, die sich auf spezifische Merkmale, wie beispielsweise "Ändere-Online-Zustand" und "Drucke-Testseite" beziehen. Es ist ferner möglich, Schnittstellen sowohl einer feineren als auch einer gröberen Granularität unter Verwendung einer vielfacher Vererbung zu unterstützen, um grobe Schnittstellen aus mehreren feinen Schnittstellen zu bilden. Die Verwendung von grobkörnigeren Schnittstellen erfordert weniger Code, wobei jedoch feinkörnigere Schnittstellen typischerweise einfacher zu verstehen sind und eine größere Erweiterbarkeit liefern.

Fig. 1 stellt ein typisches ME-Objekt mit Bezügen auf MIs dar, die eine Wiedergewinnung von Merkmalsinformationen des Geräts, das die Rolle, die durch die ME dargestellt wird, liefert, ermöglichen. Jede MI enthält eine oder mehrere Funktionen, wobei einige von diesen eine Wiedergewinnung der Informationen von dem Gerät ermöglichen, und andere von diesen ermöglichen können, daß das zugeordnete Gerät eine Funktion durchführt. Wieder sei daran erinnert, daß die ME sich auf eine Rolle bezieht, die durch ein Gerät durchgeführt wird, und nicht notwendigerweise auf das Gerät als Ganzes – es sei denn, das Gerät führt lediglich eine Rolle durch.

Das Vererbungsdiagramm von **Fig.** 2 ist ein vereinheitlichtes Modellierungssprachen-Diagramm, das zeigt, wie MIs mit MEs zusammenhängen. Die Pfeile, die den ME-Block mit den MI- und "Verwaltungsschnittstellenanbieter"-Blöcken verbinden, zeigen eine Vererbung an. Spezieller diktiert die Vererbung, daß, wenn ein Programm kompiliert wird, das ME-Objekt alle Verfahren der vererbten Schnittstellen implementiert. ME-Objekte sind sowohl Anbieter von MIs als auch MIs selbst.

Das "Verwaltungsschnittstellenanbieter"-Objekt, das in **Fig.** 2 gezeigt ist, umfaßt drei Funktionen, die durch die PSM-Software verwendet werden. Die Funktion "Besitzt-Verwaltungsschnittstelle()" bestimmt, ob eine ME einen Zeiger zu einer spezifizierten MI zurückgeben kann. Die Funktion "Erhalte-Verwaltungsschnittstelle()" gibt eine MI zurück, die verwendet werden soll, um Informationen bezüglich einer zugeordneten ME wiederzugewinnen, oder um zu veranlassen, daß die ME einige Funktionen durchführt. Die Funktion "Erhalte-Verwaltungsschnittstellenliste()" gibt eine Liste von zugeordneten MIs zurück.

In **Fig.** 3 sind die unterschiedlichen Arten von MEs ausführlicher dargestellt. Peripheriegeräte, Peripherieserver und Peripherie-Clients werden von "Druckweg-Verwaltete-Entität" abgeleitet, um deren gemeinsame Beziehung dahingehend, daß dieselben ein Teil des Druckwegs sind, zu identifizieren. **Fig.** 3 ist nicht dazu bestimmt, vollständig zu sein. Es ist möglich, andere Typen von MEs vorzusehen. Zum Beispiel kann ein Drucker einen nachgerüsteten Ausgabebehälter eines anderen Herstellers aufweisen, der Blätter faltet und in Umschläge steckt. Ein solches Gerät kann eine ME sein, die sich von der ME unterscheidet, die den Drucker darstellt.

MIs können von anderen MIs abgeleitet werden. Wenn eine abgeleitete MI durch eine ME unterstützt wird, sollte diese ME die Eltern-MIs dieser abgeleiteten MI unterstützen. Da die meisten Komponententechnologien eine Schnittstellenvererbung unterstützen, ist ein Liefern von Elternschnittstellen üblicherweise trivial.

Fig. 4 liefert ein Beispiel dafür, wie exemplarische Verwaltete-Entität-Klassen abgeleitet werden. **Fig.** 4 enthält zwei Verwaltungsschnittstellen, die nicht auch MEs sind: "Ändere-Online-Zustand" und "Drucke-Testseite". "LaserJet2000" und "LaserJet2001" müssen nicht notwendigerweise als einzigartige MEs definiert sein. Dieselben können einfach als Drucker-MEs definiert sein und liefern alle anderen MIs, die unterstützt werden.

Ort von MEs

Ein ME-Objekt kann an einem von drei Plätzen angeordnet sein:

- eingebettet in die Maschine, die dasselbe darstellt (wie anhand eines Beispiels unten beschrieben wird);
- angeordnet in einer Maschine, die direkt mit der Maschine verbunden ist, die dasselbe darstellt;
- angeordnet in einer Verwaltungsservermaschine.

Der Hauptvorteil des Einbettens von ME-Objekten ist die Beseitigung der Abhängigkeit von existierenden Protokollen. Wenn ein ME-Objekt in ein Gerät eingebettet ist, kann dasselbe mit diesem Gerät auf eine gerätespezifische Art und Weise zusammenwirken. Da es nicht zweckmäßig ist, alle existierenden Peripheriegeräte mit eingebetteten ME-Objekten nachzurüsten, können ME-Objekte auf einer Verwaltungsservermaschine angeordnet sein und können unter Verwendung von existierenden Protokollen kommunizieren. Dies ist der Hauptvorteil des Anordnens von Objekten innerhalb einer Verwaltungsservermaschine. Der Hauptvorteil des Anordnens von MEs in direkt verbundenen Maschinen besteht darin, Situationen zu bewältigen, bei denen ein Peripheriegerät nicht mit einem Netz verbunden ist. Da eine Technologie mit verteilten Komponenten eine Netzverbindbarkeit erfordert, muß eine ME auf der direkt verbundenen Maschine ein Stellvertreter für ein direkt verbundenes Gerät sein, auf das nicht durch das Netz zugegriffen werden kann.

Der Ort einer ME kann sich mit der Zeit ändern. Es sei ein Peripheriegerät mit einer eingebetteten ME, die nicht mit einer IP-Adresse konfiguriert wurde, betrachtet. Ohne eine IP-Adresse (oder eine andere Netzprotokollunterstützung) ist es nicht möglich, mit der eingebetteten ME zu kommunizieren. Um diese Situation unterzubringen, kann eine Stellvertreter-MB, die im folgenden als Proxy-ME bezeichnet wird, in der Verwaltungsservermaschine erzeugt werden. Diese Proxy-ME wird nicht alle Schnittstellen, die durch die eingebettete ME geliefert werden, unterstützen, sondern wird die Schnittstelle unterstützen, die ermöglicht, daß die IP-Adresse eingestellt wird. Nachdem die IP-Adresse eingestellt wurde, wird die eingebettete ME dem ME-Proxy, der auf der Verwaltungsservermaschine angeordnet ist, übergeordnet sein (supercede). Die "Datenbank"- und "Explorer"-Abstraktionen (die nachfolgend beschrieben werden) sind für das Erzeugen und Ersetzen der MEs verantwortlich.

"Explorer"

Definition

Ein "Explorer" ist ein Objekt, das Geräte entdeckt, die den MEs entsprechen, MEs erzeugt, die diesen Geräten entsprechen (falls erforderlich) und ME-Bezüge zu einer Datenbank, die verwendet wird, um dieselben zu verfolgen, liefert. "Explorer" haben einen Zustand und können konfiguriert werden.

"Operationen" und "Ansichten" (die unten beschrieben werden) wirken nicht direkt mit den Geräten zusammen. Stattdessen wirken diese mit weiteren Objekten, z. B. einer ME, zusammen. Dies befreit das Objekt von der Kenntnis der Einzelheiten dessen, wie mit einem besonderen Gerät kommuniziert wird, da die verschiedenen Implementierungen der ME-Abstraktion diese Kenntnis einkapseln. MEs sind entweder in das Gerät eingebettet, das dieselben darstellen oder auf einer anderen Maschine (d. h. den Proxies (Stellvertretern)) angeordnet. Eingebettete MEs werden spezialisiert (instantiated), wenn das Gerät, in dem sie sich befinden (und das sie darstellen), gebootet wird. Nicht eingebettete MEs existieren nicht, bis das Gerät, das sie darstellen, entdeckt wird. Wenn dies auftritt, wird der geeignete Typ einer ME spezialisiert. (Der geeignete Typ wird durch ein Abfragen des Geräts während der Entdeckung bestimmt.)

Unterschiedliche "Explorer" werden zum Entdecken von unterschiedlichen MEs geliefert, d. h. einer für MEs, die eine Druckerrolle durchführen, einer für MEs, die eine Scannerrolle durchführen, etc.. "Explorer" entdecken einfach Geräte, erzeugen MEs, falls erforderlich, und übergeben die ME an eine "Datenbank"-Abstraktion (die unten erörtert wird).

Es existieren mehrere hundert Typen von Geräten, die MEs entsprechen, wobei jedes Jahr weitere hinzukommen. Um dies anzugehen, kann ein "Explorer" eine Tabelle verwenden, die Gerätetypen auf ME-Erzeugungsverfahren abbildet, um zu bestimmen, wie eine ME, die einem besonderen Gerätetyp entspricht, zu erzeugen ist. Da MEs den Mechanismus für eine Kommunikation einkapseln, können unterschiedliche Typen von MEs, abhängig davon, wie ein besonderer Gerätetyp verbunden ist, erzeugt werden. Zum Beispiel kann ein direkt verbundener Laserdrucker durch einen anderen Typ einer ME dargestellt werden, als ein Laserdrucker, der durch ein Netz verbunden ist. Da ein anderer Typ eines "Explorers" verwendet wird, um direkt verbundene Geräte zu entdecken, als zum Entdecken von Netzgeräten verwendet wird, wird der "Explorer" ein anderes ME-Erzeugungsverfahren verwenden, um die ME zu erzeugen.

Obwohl oben ein spezifisches Verfahren zum Spezialisieren der MEs erörtert wurde, ist es wichtig, daß angemerkt wird, daß die Verfahren, die oben beschrieben wurden, lediglich ein Beispiel dafür sind, wie MEs spezialisiert werden können. Jeder "Explorer" ist frei, seinen eigenen Mechanismus zum Spezialisieren von MEs zu implementieren. Tatsächlich spezialisiert ein "Explorer" überhaupt keine MEs, wenn bereits existierende MEs (z. B. eingebettet in das Gerät) entdeckt werden (da diese bereits spezialisiert wurden, als das Gerät gebootet wurde).

Beispiele von "Explorern"

"Lokales-IP-Unternetz-Explorer"

Oft sind die am meisten interessierenden Geräte diejenigen Geräte, die auf einem lokalen Unternetz angeordnet sind. Diese Geräte können durch ein Netzadministrationsprogramm entdeckt werden, wobei jedoch statt eines Erzeugens von Aufzeichnungen für die Programm-"Datenbank" eine ME des geeigneten Typs erzeugt wird.

"Direktverbindungs-Explorer"

Ein "Explorer" kann erzeugt werden, der alle lokalen Tore nach direkt verbundenen Geräten untersucht, und, basierend auf seinen Ergebnissen, den geeigneten Typ von MEs als Stellvertreter für direkt verbundene Geräte erzeugt.

"Server-Explorer"

Ein "Explorer" kann erzeugt werden, der die MEs von einem Server "entdeckt". Ein solcher "Explorer" wird in Umgebungen verwendet, in denen ein Server, der routinemäßig Entdeckungen durchführt, bereits existiert. Dies kann verwendet werden, um den Netzverkehr zu reduzieren und das Verhalten zu verbessern.

"Verwaltete-Entität-Explorer"

Es ist nicht geeignet, jede ME auf einem Netz zu entdecken. Viele MEs werden Proxies sein, die auf dem Verwaltungsserver angeordnet sind und dazu bestimmt sind, lediglich für die Verwaltungsanwendung auf dem Verwaltungsserver verwendet zu werden. Jedoch können andere mit der Absicht in ein Peripheriegerät eingebettet sein, daß diese MEs durch eine entfernte Peripherieverwaltungssoftware verwendet werden. Die PSM-Software kann diejenigen MEs entdecken, die geeigneterweise entdeckt werden sollen, ohne MEs zu entdecken, die geeigneterweise nicht entdeckt werden sollen, unter Verwendung eines bereits existierenden Benennungsdienstes, z. B. eines CORBA-Benennungsdienst (Common Object Request Broker Architecture = gemeinsame Objektanforderungsvermittlungsarchitektur). Da MEs selbst wissen, ob auf sie extern zugegriffen werden soll oder nicht, können diese sich bei dem Benennungsdienst registrieren, was ermöglicht, daß dieselben auffindbar sind. Diese Lösung kann erfordern, daß ein einzelner Benennungsdienst auf dem Netz eingerichtet wird. Eine weitere Lösung besteht darin, unter Verwendung eines weiteren Protokolls (z. B. SNMP = Simple Network Management Protocol-einfaches-Netzverwaltungsprotokoll)-Geräte-zu-entdecken-und-ein-Attribut-des-Protokolls zu verwenden, um die eingebettete ME des Geräts zu identifizieren. Obwohl jedes zugrundeliegende Protokoll einen unterschiedlichen "Explorer" benötigt, um die Entdeckung von eingebetteten MEs zu unterstützen, könnte die Bereitstellung eines Benennungsdienstes vermieden werden.

"Operationen"

Definition

Eine "Operation" ist ein Objekt, das bezüglich einer Ansammlung von MEs unter Verwendung von MIs wirksam ist. Jede ME in der Ansammlung, die bearbeitet wird, muß die MIs unterstützen, die durch die Operation benötigt werden. "Operationen" haben einen Zustand und können konfiguriert sein.

Wenn neue MIs definiert sind, können neue "Operationen" definiert werden, die diese MIs verwenden. PSM-Anwendungen können einfach "Operationen" im allgemeinen – ohne spezifische Kenntnis einer besonderen "Operation" – unterstützen. Eine "Operation" benötigt bestimmte MIs von den MEs, bezüglich derer dieselbe wirksam ist. Eine "Operation" kann mehr als eine MI benötigen, benötigt jedoch mindestens eine.

Abhängig von der "Operation" kann die Reihenfolge der MEs in einer Ansammlung signifikant sein oder nicht. Zum Beispiel ist die Anzahl signifikant für eine "Operation", die verwendet wird, um IP-Adressen einer Ansammlung von MEs zuzuweisen. Die Reihenfolge ist jedoch unwichtig für eine "Operation", die verwendet wird, um Firmware nachzurüsten.

"Operationen" können abhängig von der "Operation" konfigurierbar sein oder nicht. Eine "Operation", die lediglich den Zustand auf "online" ändert, erfordert keine Konfiguration. Bei "Operationen", die konfigurierbar sind, wird ein Applet vorgesehen, das weiß, wie dieser Typ einer "Operation" konfiguriert werden soll. Ein Applet ist verantwortlich für das Liefern einer Benutzerschnittstelle, die ermöglicht, daß ein Zustand modifiziert wird. Ein solches Applet wird auf der Verwaltungs-Client-Maschine laufen und die "Operation" durch ein Aufrufen der Verfahren konfigurieren. Bei einer bevorzugten Implementation ist das Applet in Java geschrieben.

Beispiele von "Operationen"

"Online-Operation"

Es sei an MI "Ändere-Online-Zustand", die in dem Klassenhierarchiediagramm von **Fig. 4** gezeigt ist, erinnert. Diese MI ermöglicht, daß der Online-Zustand einer ME, die diese MI unterstützt, geändert wird. Eine "Operation" ist geschaffen, die die MI "Ändere-Online-Zustand" verwendet, um sicherzustellen, daß der Online-Zustand "online" ist. Die Online-"Operation" ist zum Erzeugen einer Verwaltungsanwendung verwendbar, die periodisch alle Drucker auf "online" einstellt.

"Installiere-Schriftart-Operation"

Man betrachte die folgende Schnittstelle, die verwendet wird, um True-Typ-Schriftarten zu verwalten:

Schnittstelle "True-Type-Schriftartverwaltung"

```
ManagementInterface
{
      void Add( Font f);
      void FontList List();
      void Remove(FontListElement f);
};
```

Man betrachte nun eine Peripheriesystemverwaltungsanwendung, die Schriftarten auf Endbenutzermaschinen (z. B. PCs) und Druckern installiert. Eine solche Anwendung kann unter Verwendung einer "Operation" aufgebaut werden, die eine Schriftart auf einer Maschine installiert, die die Schnittstelle "True-Type-Schriftartverwaltung" unterstützt. Die Anwendung identifiziert einfach alle MEs, die die Schnittstelle "True-Type-Schriftartverwaltung" unterstützen, konfiguriert die "Operation" "Installiere-Schriftart" mit der erwünschten Schriftartdatei und ruft die "Operation" bei hinsichtlich der MEs, die vorher identifiziert wurden, auf.

"Zuweisen-Sequentieller-Netzadressen-Operation"

Wenn ein physisches Netz von einem Unternetz in ein anderes geändert wird, ist es erforderlich, alle Geräte in dem physischen Netz neu zu numerieren. Netzadministrationsprogramme unterstützen Mechanismen zum Neuzuweisen von Netzadressen nacheinander zu allen Geräten (die durch solche Anwendungen verwaltet) auf dem physischen Netz. Dies kann unter Verwendung einer "Operation" erreicht werden. Eine solche "Operation" ist mit einem Bereich von Netzadressen konfiguriert. Die Peripheriegeräte, die neu numeriert werden sollen, werden dann durch die Anwendung ausgewählt (mit oder ohne Benutzereingriff, abhängig von der Anwendung). Schließlich wird die "Zuweisen-Sequentieller-Netzadressen-Operation" auf der vorher ausgewählten Ansammlung der Peripheriegerät-MEs auf gerufen.

"Ansichten"

Definition

Eine "Ansicht" ist ein Objekt, das eine Ansammlung von MEs anzeigt, die MIs verwenden. Jede ME in der Ansammlung, die angezeigt wird, muß die MIs, die durch die "Ansicht" benötigt werden, unterstützen. "Ansichten" haben einen Zustand und können konfiguriert werden.

Eine "Ansicht" ist einer "Operation" sehr ähnlich, mit der Ausnahme, daß statt eines Arbeitens bezüglich der gelieferten MEs die MEs angezeigt werden. Eine "Ansicht" kann in drei Teile geteilt werden: einen Zustand, eine Konfiguration und eine Anzeige. Der Zustand verfolgt die gegenwärtige Konfiguration und hat keine Benutzerschnittstelle. Die Konfiguration ist ähnlich einem Konfigurations-Applet für eine "Operation". Diese ist für ein Darstellen einer Benutzerschnittstelle verantwortlich, die ermöglicht, daß der Zustand modifiziert wird. Die Anzeige ist für ein Anzeigen einer Ansammlung von MEs verantwortlich. Die Reihenfolge der MEs kann, abhängig von der "Ansicht", wichtig sein oder nicht. Einige "Ansichten" können die Reihenfolge der MEs in einer Ansammlung ignorieren und einfach die obersten drei "interessierenden" MEs anzeigen (und den Rest zusammenfassen). Andere können alle MEs in der Reihenfolge, in der sie in der Ansammlung geliefert werden, anzeigen. Mehr als ein Fall einer "Ansicht" kann zur selben Zeit existieren, mit einem unterschiedlichen Zustand.

Beispiele von "Ansichten"

"Verbrauchsmaterial-Ansicht"

Es ist kostspielig, wenn bei einem Drucker der Toner ausgeht. Da Benutzer oft untrainiert sind, dauert das Wechseln einer Tonerkassette länger, als wenn ein Techniker die Kassette ersetzt. Untrainierte Benutzer können ferner versehentlich den Drucker beschädigen. Um diesem Problem zu begegnen, ist es nützlich, eine Einrichtung zu besitzen, die herausfindet, welche Drucker fast keinen Toner mehr haben, so daß der Toner ersetzt werden kann, bevor derselbe erschöpft ist. Dies wird unter Verwendung einer "Ansicht" erreicht. Eine solche "Ansicht" zeigt Drucker unter einer gewissen Tonerschwelle an, der Reihe nach von einer geringsten bis zu einer höchsten Tonermenge. Durch ein Ändern der Niedrigtonerschwelle kann der Benutzer die Anzahl von Druckern, die angezeigt werden, variieren. Alle Drucker werden angezeigt, wenn die Niedrigtonerschwelle auf 100% gesetzt ist.

"Scanner-Ansicht"

Diese "Ansicht" zeigt MEs eines spezifischen Typs an. Eine "Ansicht" kann für Drucker erzeugt werden und würde die maximale Zahl der Seiten pro Minute zeigen, die der Drucker liefern kann. Eine "Ansicht" für Scanner kann ein Feld haben, das anzeigt, ob der Scanner Farbe unterstützt. Demgemäß können "Ansichten" erzeugt werden, um spezifische Typen von MEs zu unterstützen und Felder anzuzeigen, die für diesen Typ sinnvoll sind.

"Abbildungs-Ansicht"

Eine "Ansicht" kann aufgebaut werden, die, wenn diese mit einer Abbildung konfiguriert ist, MEs gemäß deren Ort auf der Abbildung anzeigt. Der Ort der MEs wird von einer Ortsdatei abgeleitet, die den Zustand der Abbildungs-"Ansicht" umfaßt. Da "Ansichten" einen Zustand haben, kann dieser Zustand verwendet werden, um Informationen, die aus Interaktionen zwischen dem Benutzer und der Ansicht resultieren, zu speichern.

"Datenbanken"

Definition

Eine "Datenbank" ist ein Objekt, das MEs annimmt, Duplikate handhabt und MEs für eine spätere Verfügbarkeit speichert. Eine "Datenbank" kann ferner von einer besonderen ME oder allen MES gelöscht werden.

Die "Datenbank"-Abstraktion ermöglicht, daß andere Architekturabstraktionen (wie beispielsweise "Explorer") mit vielen Typen von "Datenbanken" zusammenwirken, ohne den spezifischen Typ der "Datenbank" zu kennen. Eine Anwendung, die den spezifischen Typ der "Datenbank", die verwendet wird, kennt, ist weiterhin frei, um die volle Funktionalität jedes verwendeten zugrundeliegenden "Datenbank"-Verwaltungssystems zu verwenden. Das Ziel der "Datenbank"-Abstraktion besteht darin, ein Objekt zu liefern, mit dem "Explorer" zusammenwirken können und das eine Kommunikation mit dem Speichermechanismus der Anwendung ermöglicht.

Beispiele von "Datenbanken"

"Dateibasierte Datenbank"

Bei einigen Anwendungen kann das einfache Besitzen einer Datei, die eine Liste der MEs enthält, akzeptabel sein. Eine dateibasierte "Datenbank" kann durch ein Implementieren der Schnittstelle, die für die "Datenbank"-Abstraktion spezifiziert ist, erzeugt werden ("Hinzufügen", "Entfernen Einer", "Entfernen Aller"), so daß Informationen, die sich auf MEs beziehen, in einer Datei aufgezeichnet werden. Wenn zu der "Datenbank" eine ME hinzugefügt wird, kann eine Zeile zu der Datei mit den Informationen, die durch die Anwendungen benötigt werden, sowie mit Informationen, die erforderlich sind, um die ME eindeutig zu identifizieren, hinzugefügt werden (um "Entfernen-Einer"- und "Handhabungs"-Duplikationen zu unterstützen).

"Objektorientierte Datenbank"

Bei einigen Anwendungen kann die Flexibilität, die durch eine objektorientierte "Datenbank" geliefert wird, attraktiv sein. Eine solche "Datenbank" kann ähnlich zu der implementiert werden, die für eine dateibasierte "Datenbank" implementiert ist (lediglich am Anfang einer objektorientierten "Datenbank").

"Relationale Datenbank"

Bei einigen Anwendungen, die ein Datenbankverhalten benötigen, kann es attraktiv sein, eine relationale "Datenbank" zu verwenden. Eine solche "Datenbank" kann ähnlich zu der implementiert werden, die für eine dateibasierte "Datenbank" implementiert ist (lediglich am Anfang einer relationalen "Datenbank").

Systemkonfiguration

Bezugnehmend auf **Fig. 5** wird ein System zum Implementieren der Erfindung dargestellt. Ein Client-Computer **10** umfaßt eine Zentral-Verarbeitungs-Einheit (CPU) **12**, ein Netzverbindungsmodul **14** und einen Speicher **16**, der ein PSM-Objekt **18** hält. Das PSM-Objekt **18** umfaßt den notwendigen Code, um die verschiedenen Objekte zu verwenden, die oben beschrieben wurden, um eine Mehrzahl von Peripheriegeräten zu verwalten, auf die durch ein Netz **20** zugegriffen werden kann.

Jedes der Peripheriegeräte **22** und **24** umfaßt ein "Verwaltungsschnittstellenanbieter-Objekt" 26, das wiederum einen Zugriff auf ein "Besitzt-Schnittstellen-Verfahren" 28, "Schnittstellen-Auflisten-Verfahren" 30 und ein "Schnittstellen-Erhalten-Verfahren" 32 ermöglicht. Jedes Peripheriegerät umfaßt ferner ein oder mehrere eingebettete ME-Objekte, wobei jedes einen Bezug auf eine oder mehrere MI(s) ermöglicht, die eine Wiedergewinnung von Informationen bezüglich einer jeweiligen Peripheriegerätrolle ermöglichen. Es sei in Erinnerung gerufen, daß jede MI ein Objekt ist, das mindestens eine Funktion umfaßt, deren Ausführung die Wiedergewinnung von Informationen bezüglich einer Rolle, die durch das jeweilige Peripheriegerät durchgeführt wird, ermöglicht.

Hierin anschließend wird angenommen, daß alle Objekte bereits in den Speicher geladen sind und für eine Verwendung bereit sind. Jedoch liegt es im Schutzbereich dieser Erfindung, solche Objekte auf einem oder mehreren Datenspeichergeräten zu liefern, wie beispielsweise einer Magnetplatte **29**, wobei der Inhalt dieser auf einer Bedarfsbasis in den Speicher geladen werden kann.

Der Speicher **16** umfaßt ferner ein oder mehrere "Explorer"-Objekte **34**, von denen jedes ein Verfahren zum Erhalten von Bezügen auf ME-Objekte enthält; "Ansicht"-Objekte **36**, die das Betrachten von ausgewählten MEs, Zuständen und Konfigurationen ermöglichen (wie oben beschrieben ist); "Operations"-Objekte **38**, die bezüglich einer Ansammlung von MEs unter Verwendung von MIs wirksam sind (wie oben beschrieben ist); und ein "Datenbank"-Objekt **40** (wie oben beschrieben ist).

Beispielanwendung

"Aktualisiere-Firmware"

Um die "Operation" des PSM-Verfahrens **18** besser zu verstehen, wird auf die **Fig.** 6 und 7 Bezug genommen, die ein logisches Flußdiagramm des Verfahrens der Erfindung umfassen, das eine Anwendung darstellt, die für eine Nachrüstung der Firmware der Netzdrucker verantwortlich ist. Die Anwendung führt die folgenden Aktivitäten durch:

1. Lokalisieren aller verfügbaren Drucker auf dem Netz.
2. Anzeigen einer Liste von Druckern, deren Firmware nachgerüstet werden kann, was durch ein Druckermodell organisiert wird.
3. Ermöglichen, daß der Benutzer einen oder mehrere Drucker von der Liste von Druckern auswählt.
4. Nachrüsten der Firmware auf jedem der ausgewählten Drucker.

Hintergrund

Drucker enthalten Software, die verwendet wird, um Druckjobs zu verarbeiten, die Firmware genannt wird (da sich dieselbe in dem ROM oder einem anderen halbpermanenten Speicher, wie beispielsweise einem Flash-Speicher befindet). Es ist manchmal wünschenswert, die Druckerfirmware wie die Software auf einem PC zu aktualisieren, um Defekte zu reparieren oder neue Merkmale hinzuzufügen. Es ist wünschenswert, daß die Verwalter eine Möglichkeit besitzen, die Firmware aller Drucker des gleichen Modells zu aktualisieren, und nicht jeden Drucker als eine einzelne Handlung manuell zu aktualisieren.

Ausführung

1. Die Anwendung "Aktualisiere-Firmware" startet (Schritt **100**, **Fig.** 6). An diesem Punkt hat die Anwendung keine MEs.
2. Die Anwendung lokalisiert MEs unter Verwendung von einem oder mehreren "Explorern". Die Anwendung besitzt eine Liste von bekannten "Explorern", die beim Hochfahren aufgebaut wird. Diese Liste kann auf mehrere Arten aufgebaut werden, wobei jedoch der Einfachheit halber angenommen wird, daß die Liste durch ein Lesen einer Datei erhalten wird. Man nehme ferner an, daß sich jeder "Explorer" in einer getrennten, dynamisch verbundenen Bibliothek (DLL; DLL = Dynamic Linked Library) befindet und die Liste den Ort in dem Dateisystem von DLLs identifiziert, die den ausführbaren Code für jedes "Explorer"-Objekt enthalten. Man nehme ferner an, daß die Liste von "Explorern" lediglich einen Eintrag hat. Die Anwendung liest die Liste, lädt die DLL (unter Verwendung von Betriebssystemaufrufen) und ruft eine exportierte Funktion der DLL auf, um einen Fall des "Explorer"-Objekts zu spezialisieren (Schritt **102**).
3. Nun hat die Anwendung ein "Explorer"-Objekt. Der nächste Schritt besteht darin, das "Explorer"-Objekt zu verwenden, um eine Liste von MEs zu erhalten, was ein Aufrufen des "Explorer"-Objekts erfordert. Das Verfahren des "Explorer-"-Objekts gibt eine Datenstruktur, wie beispielsweise ein Array, das eine Liste von entdeckten MEs liefert, zurück. Die folgende abstrakte Basisklasse in C++ demonstriert eine mögliche Schnittstelle für einen "Explorer" (und insbesondere die Schnittstelle, die bei diesem Beispiel verwendet wird):

```
class Explorer {
    virtual void explore(ManagedEntity *MEList,
                         int           *MEListLength) =
    0;
};
```

4. Die obige abstrakte Basisklasse liefert die Schnittstelle eines "Explorers", beschreibt jedoch nicht die Implementierung. Da sehr viele Implementierungen existieren können, wird der Einfachheit halber angenommen, daß der "Explorer" einen Bezug auf eine Textdatei enthält, die eine Liste von MEs, die in Drucker eingebettet sind, enthält. (Jede dieser MEs wurde erzeugt, als der Drucker eingeschaltet wurde.) Jedes der ME-Objekte von der Textdatei wird zu der aufrufenden Anwendung zurück geleitet.
5. Nach dem Aufrufen des Erforschungsverfahrens auf dem "Explorer"-Objekt hat die Anwendung nun eine Ansammlung von MEs in einem Array. Der nächste Schritt besteht darin, solche MEs anzuzeigen, die eine MI "Fern-Firmware-Aktualisierung" liefern (Schritt **104**). Um dies zu erreichen, muß die Anwendung bestimmen, ob jede ME die MI "Fern-Firmware-Aktualisierung" liefert oder nicht. Dies wird durch ein Aufrufen eines Verfahrens "Besitzt-Schnittstelle" 28 bezüglich jeder der MEs erreicht. Zum Beispiel kann die Schnittstelle einer MI durch die folgende abstrakte Basisklasse von C++ dargestellt werden:

```
class ManagedEntity {
    boolean hasManagementInterface(string mi_name) = 0;
    ManagementInterface *getManagementInterface (string
    mi_name) = 0;
    void getManagementInterfaceList(string *strList,
                                    int    *strListLen)
    = 0;
};
```

6. Wenn "Besitzt-Schnittstelle" aufgerufen wird, muß jede ME "wahr" oder "falsch" zurückgeben (Schritt **106**). Die Antwort, die jede ME liefert, wird durch ein Konsultieren einer internen Tabelle bestimmt, die jeden der MI-Namen auflistet, die durch die ME unterstützt werden. Die Funktion "Besitzt-Schnittstelle" vergleicht einfach den gelieferten MI-Namen mit jedem der MI-Namen, die in der Tabelle enthalten sind. Wenn keine Übereinstimmung gefunden wurde, gibt "Besitzt-Schnittstelle" ein "falsch" zurück. Falls eine Übereinstimmung gefunden wurde, gibt "Besitzt-Schnittstelle" ein "wahr" zurück. Obwohl die obige Beschreibung MEs als eine Tabelle mit Verwaltungsschnittstellennamen aufweisend darstellt, ist dies nicht notwendigerweise die Art, auf die alle MEs implementiert werden.

7. Sobald jede der MEs abgefragt wurde, um zu bestimmen, ob dieselbe die Verwaltungsschnittstelle "Fern-Firmware-Nachrüstung" unterstützt, besteht der nächste Schritt darin, solche MEs anzuzeigen, die diese Schnittstelle unterstützen. Die Anwendung könnte direkt Informationen über diese MEs anzeigen, wobei es jedoch flexibler ist, diese MEs unter Verwendung eines "Ansicht"-Objekts anzuzeigen. Ein "Ansicht"-Objekt ist wiederverwendbar und kann bei anderen Anwendungen als der Anwendung, die gegenwärtig erörtert wird, verwendet werden. Ein "Ansicht"-Objekt erfordert eine oder mehrere besondere MEs, um zu funktionieren. In diesem Fall wird das -"Ansicht"-Objekt die MI "Fern-Firmware-Nachrüstung" benötigen.

Die Verwaltungsschnittstelle "Fern-Firmware-Nachrüstung" kann durch die folgende abstrakte Basisklasse dargestellt werden:

```
class RemoteFirmwareUpgrade : ManagementInterface {
    unsigned long firmwareVersion version() = 0;
    void upgradeFirmware( string newFirmwareURL ) = 0;
    string printerModel() = 0;
};
```

Obwohl bei einer tatsächlichen Implementierung die Funktion "Drucker-Modell" ein Teil einer anderen allgemeineren MI sein würde, ist dieselbe in der MI "Fern-Firmware-Nachrüstung" enthalten, um dieses Beispiel zu vereinfachen. Die MI "Fern-Firmware-Nachrüstung" enthält alle Funktionen, die erforderlich sind, um dieses Beispiel zu vollenden. Vor einem Anzeigen der MEs, die die MI "Fern-Firmware-Nachrüstung" unterstützen, ist es notwendig, ein "Ansicht"-Objekt zu spezialisieren (Schritt **108**).

8. Das Spezialisieren eines Ansicht-Objekts wird (bei diesem Beispiel) durch ein Laden einer DLL in den Speicher unter Verwendung von Betriebssystem-APIs und ein Aufrufen eines Verfahrens, um das "Ansicht"-Objekt zu erzeugen, erreicht.

9. Sobald das "Ansicht"-Objekt erzeugt ist, kann dasselbe verwendet werden, um die MEs anzuzeigen. Eine "Ansicht" kann als die folgende Schnittstelle, die als eine abstrakte Basisklasse von C++ definiert ist, aufweisend aufgefaßt werden:

```
class ManagementView {
    void display(ManagedEntity *me_list,
                 int           *me_listLen) = 0;
    void getSelectedMEs(ManagedEntity *me_list,
                        int           *me_listLen) = 0;
};
```

Um eine Ansammlung von MEs anzuzeigen, wird das Anzeigeverfahren mit einem Array von ME-Zeigern aufgerufen. Ansprechend hierauf führt das "Ansicht"-Objekt folgende Schritte aus:

- Verifizieren, daß jede ME, die in dem ME-List-Array geliefert wird, die MI "Fern-Firmware-Nachrüstung" durch ein Aufrufen des Verfahrens "Besitzt-Schnittstelle" bezüglich jeder ME unterstützt. Ansprechend darauf wird, wie vorher beschrieben wurde, jede ME eine interne Datenstruktur konsultieren, um zu bestimmen, ob die gelieferte MI unterstützt wird. Wenn eine ME, die die MI "Fern-Firmware-Nachrüstung" nicht unterstützt, in dem ME List-Array existiert, tritt ein Fehler auf.

– Erhalten der tatsächlichen MI "Fern-Firmware-Nachrüstung" von jeder ME (Schritt **108**). Dies wird durch ein Aufrufen der Funktion "Erhalte-Verwaltungsschnittstelle" bezüglich jeder ME mit dem Namen der erwünschten MI erreicht, der in diesem Fall "Fern-Firmware-Nachrüstung" ist. Jede MI ist als ein Zeiger auf eine MI-Klasse dargestellt. Man erinnere sich, daß sich die tatsächlichen MEs entfernt in Druckern befinden. Dies bedeutet, daß die Implementierungen der Funktionen, die die MIs bilden, sich ebenfalls entfernt in dem Drukker befinden. Obwohl eine Client-Software, wie beispielsweise das "Ansicht"-Objekt, mit lokalen Schnittstellen (z. B. "Fern-Firmware-Nachrüstung") zusammenwirken, verwenden diese lokalen Schnittstellen eingerichtete Kommunikationsverfahren, um Parameter und Verfahren von den tatsächlichen Implementierungen, die sich in dem Drucker befinden, zusammenszustellen. Sobald diese erhalten werden, werden die MIs typmäßig als MIs "Fern-Firmware-Nachrüstung" eingeteilt und in einer Tabelle gespeichert. Diese Tabelle enthält einen lokalen Zeiger auf die ME, einen lokalen Zeiger auf die MI "Fern-Firmware-Nachrüstung" für die entsprechende ME, eine Versionsnummer (die unten erörtert wird) und ein Modell (das unten erörtert wird).

– Erhalten der Versionsnummer jeder ME durch ein Aufrufen des Verfahrens "Firmware-Version", das durch die MI "Fern-Firmware-Nachrüstung" geliefert wird (jede ME hat ihre eigene unabhängige Implementierung dieser MI). Der Wert, der erhalten wird, wird in der Tabelle gespeichert, die verwendet wird, um Informationen für jede ME zu speichern (wie beispielsweise Schnittstellenzeiger).

– Erhalten des Druckermodells jeder ME durch ein Aufrufen der Funktion "Drucker-Modell", die durch die MI "Fern-Firmware-Nachrüstung" geliefert wird (jede ME weist ihre eigene unabhängige Implementierung dieser MI auf). Der Wert, der erhalten wird, wird in der Tabelle gespeichert, die verwendet wird, um Informationen für jede ME zu speichern (wie beispielsweise Schnittstellenzeiger).

10. Da die Tabelle, die verwendet wird, um Informationen für jede ME zu speichern, nun bevölkert ist, kann die Tabelle durch das Druckermodell und die Firmwareversion sortiert werden. Ein Sortieren kann durch jede von mehreren eingerichteten Einrichtungen erreicht werden. Das "Ansicht"-Objekt kann dann ein Fenster öffnen und die sortierte Liste der ME-Informationen anzeigen. In dieser Liste kann dasselbe spezifisch das Druckermodell und die Firmwareversion zeigen. Wenn das "Ansicht"-Fenster angeklickt wird, bestimmt das "Ansicht"-Objekt, welcher ME-Eintrag in der Tabelle angeklickt wurde und führt zwei Dinge aus. Erstens hebt das "Ansicht"-Objekt diesen Eintrag in dem Fenster hervor. Zweitens modifiziert das Ansichtobjekt ein Feld in der ME-Tabelle, um anzuzeigen, daß das Feld ausgewählt ist.

Wie oben angezeigt ist, wurde die Tabelle, die verwendet wird, um die MEs innerhalb des "Ansicht"-Objekts zu verfolgen, derart beschrieben, daß dieselbe Felder für einen lokalen Bezug auf die ME, einen lokalen Bezug auf eine MI "Fern-Firmware-Nachrüstung", einen Modellnamen und eine Firmwareversionsnummer, aufweist. Diese Tabelle wird erweitert, um ein neues boolsches Feld zu umfassen, das anzeigt, ob ein neuer besonderer Eintrag durch die Benutzerschnittstelle ausgewählt wurde.

11. Sobald der Benutzer die erwünschten MEs für eine Nachrüstung ausgewählt hat (d. h. Drucker) (Schritt **110**), kann der Benutzer durch eine Benutzerschnittstelle, die durch die Anwendung geliefert wird, wie beispielsweise ein Menüsymbol, anzeigen, daß eine Firmwarenachrüstung durchgeführt werden soll. Dies kann durch ein Herunterziehen eines Menüs, das "Operationen" genannt wird, und Auswählen von "Nachrüsten ferner Firmware" erreicht werden. "Operationen" sind Handlungen, die bezüglich einer Ansammlung von MEs durchgeführt werden. Bevor "Operationen" verwendet werden können, müssen jedoch zwei Dinge geschehen. Erstens müssen dieselben spezialisiert werden. Zweitens müssen dieselben konfiguriert werden. Die folgende abstrakte Basisklasse von C++ liefert eine mögliche Schnittstelle für "Operationen":

```
class ManagementOperation {
    void configure() = 0;
    void operate(ManagedEntity *me_list,
                 int           *me_listLen) = 0;
};
```

12. Das Erzeugen einer "Operation" kann auf mehrere Arten erfolgen, d. h. ähnlich zu der Art, auf die "Ansichten" und "Explorer" erzeugt werden. Um den Sachverhalt zu vereinfachen, wird die "Operation" durch ein Laden einer DLL in einen Speicher unter Verwendung von Betriebssystemaufrufen und Aufrufen einer Funktion, die durch diese DLL exportiert wird, um die Operation zu spezialisieren, erzeugt.

13. Sobald die "Operation" erzeugt ist, muß diese konfiguriert werden. Dies wird durch ein Aufrufen des "Konfigurieren"-Verfahrens, das durch das "Operation"-Objekt geliefert wird, erreicht. Dieses Verfahren zeigt einen Konfigurationsdialog an, der dem Benutzer ermöglicht, das "Operation"-Objekt zu konfigurieren. Die Benutzerschnittstelle enthält einen Texteditierblock mit einer Beschriftung in dessen Nähe, die anzeigt, daß der Texteditierblock den Ziel-URL (URL = Unified Resources Locator = Einheitliche-Betriebsmittel-Lokalisierer) enthält, in dem Drukkerfirmware, die nachgerüstet werden soll, gespeichert ist. Der Benutzer gibt eine Zeichenfolge ein, die einen gültigen URL dorthin liefert, wo die Druckerfirmware angeordnet ist. Sobald der Benutzer den URL eingegeben hat, ist das "Operation"-Objekt konfiguriert und betriebsbereit.

14. Als erstes muß die Anwendung die Liste von ausgewählten MEs erhalten, so daß dieselbe diese MEs zu dem "Operation"-Objekt leiten kann (Schritt **112**). Um dies zu tun, ruft die Anwendung das Verfahren "Erhalte-Ausgewählte-MEs" von dem "Ansicht"-Objekt auf. Dieses Verfahren durchläuft die Tabelle der MEs und plaziert jede ME, die ausgewählt ist (was aus dem ausgewählten Feld in der ME-Tabelle bestimmt wird), in einem Array. Dieses Array wird dann zu der Anwendung zurückgegeben. Da die Anwendung nun die ausgewählten MEs in einem Array hat, kann dieselbe das Verfahren des "Operation"-Objekts aufrufen.

15. Das Verfahren führt folgende Teilschritte aus (Schritt **114**):
- Verifizieren, daß jede der MEs, die zu demselben geleitet wird, eine MI "Fern-Firmware-Nachrüstung" besitzt, durch ein Aufrufen der Funktion "Besitzt-Schnittstelle" jeder ME.
- Erhalten eines lokalen Zeigers auf die MI "Fern-Firmware-Nachrüstung" durch ein Aufrufen des Verfahrens "Erhalte-Schnittstelle" für jede ME. Dies wird durch ein Leiten des MI-Namens der MI "Fern-Firmware-Nachrüstung" zu dem Verfahren "Erhalte-Schnittstelle" erreicht. Nach dem Verifizieren, daß alle gelieferten MEs die MI "Fern-Firmware-Nachrüstung" liefern, wird jede ME einzeln verarbeitet, bevor zu der nächsten ME gesprungen wird. Mit anderen Worten heißt das, daß der nächste Schritt abgeschlossen und dann das Verarbeiten der ME begonnen wird.
- Aufrufen des Verfahrens "Nachrüstung-Firmware" nach einem Erhalten der MI "Fern-Firmware-Nachrüstung". Das Verfahren "Nachrüstung-Firmware" wird als ein Teil der MI "Fern-Firmware-Nachrüstung" geliefert. Dieses Verfahren nimmt eine Zeichenfolge an, die den URL beschreibt, an dem die Firmware angeordnet ist.

16. Wie oben angemerkt wurde, werden sowohl MEs als auch MIs durch lokale Proxy-Objekte dargestellt (d. h. die abstrakten Basisklassen, die bei diesem Beispiel verwendet werden). Diese Klassen liefern ein vorderes Ende für ein Kommunikationsprotokoll. Die tatsächliche Implementierung des Verfahrens "Nachrüstung-Firmware" ist in jedem Drucker angeordnet (man erinnere sich, daß jeder Drucker eine ME, die in denselben eingebettet ist, enthält). Wenn das Verfahren "Nachrüstung-Firmware" aufgerufen wird, führt die Implementierung in jedem Drucker die erforderlichen Handlungen durch, um die Firmware von dem Server, der durch den URL spezifiziert ist, herunterzuladen (Schritt **116**). Sobald das Firmware-Bild heruntergeladen wurde, wird der Drucker das Firmware-Bild durch ein Überprüfen der Gültigkeit der Prüfsumme verifizieren (oder durch andere Techniken, die geeignet sind) und das Firmware-Bild in einen halbpermanenten Speicher schreiben. Schließlich wird der Drucker sich selbst neu einstellen (Warmneustart), um mit der neuen Firmware zu laufen. Sobald jede ME verarbeitet wurde und alle Firmware-Bilder nachgerüstet wurden, wird die Anwendung beendet.

Patentansprüche

1. System zum Ermöglichen, daß ein Verwaltungscomputer (**10**) Einheiten (**22, 24**) eines Peripheriesystems verwaltet, wobei jede verwaltete Einheit einen oder mehrere Dienste für einen Computer liefert, der der Verwaltungscomputer (**10**) sein kann oder nicht, und ferner ein Verwaltungsschnittstellenanbieter-Objekt speichert, das ermöglicht, daß Verfahren bezüglich Schnittstellenobjekten durchgeführt werden, wobei jeder Dienst eine zugeordnete Verwaltete-Entität-Schnittstelle (ME-Schnittstelle) (**33**) aufweist, die einen oder mehrere Bezüge auf eine oder mehrere Verwaltungsschnittstellen (MIs) umfaßt, wobei jede MI ein oder mehrere Verfahren aufweist, um zumindest (i) Informationen bezüglich eines oder mehrerer Merkmale des Dienstes, der durch die ME (**33**) dargestellt wird und durch eine verwaltete Einheit (**22, 24**) durchgeführt wird, zu liefern, oder (ii) zu bewirken, daß die verwaltete Einheit (**22, 24**) eine Funktion durchführt, wobei der Computer (**10**) folgende Merkmale aufweist:
eine Eingabe/Ausgabe-Einrichtung (**14**) zum Ermöglichen von Kommunikationen mit jeder der verwalteten Einheiten (**22, 24**);
einen Speicher (**16**) zum Speichern eines Explorer-Objekts (**34**); und
eine Verarbeitungseinrichtung (**12**), die auf eine Benutzerabfrage bezüglich eines Verwaltete-Einheit-Diensts anspricht, um (i) das Explorer-Objekt (**34**) aufzurufen, um eine ME (**33**), die dem Verwaltete-Einheit-Dienst entspricht, zu bestimmen, und (ii) ein Verfahren des Schnittstellenanbieter-Objekts (**26**) aufzurufen, um ein MI-Objekt wiederzugewinnen, das der ME (**33**) zugeordnet ist, die einen Zugriff auf Informationen, die sich auf die verwaltete Einheit (**22, 24**) beziehen, ermöglicht, oder die bewirkt, daß die verwaltete Einheit (**22, 24**) fallabhängig eine Funktion durchführt, wobei sich die Informationen oder die Funktion auf die Benutzerabfrage beziehen.

2. System gemäß Anspruch 1, bei dem der Speicher (**16**) ferner ein Ansicht-Objekt (**36**) umfaßt und der Prozessor (**12**) ein Verfahren aufruft, das dem Ansicht-Objekt (**36**) zugeordnet ist, um MEs (**33**) anzuzeigen, wobei mindestens eine der MEs (**33**) dem Verwaltete-Einheit-Dienst entspricht, wodurch ermöglicht wird, daß ein Benutzer vor dem Aufrufen einer Funktion des Schnittstellenanbieter-Objekts (**26**) eine oder mehrere der MEs (**33**) auswählt.

3. System gemäß Anspruch 1 oder 2, bei dem der Speicher (**16**) ferner ein Ansicht-Objekt (**36**) umfaßt und der Prozessor (**12**) ein Verfahren aufruft, das dem Ansicht-Objekt (**36**) zugeordnet ist, um einen Zustand einer Gruppe von MEs anzuzeigen.

4. System gemäß einem der Ansprüche 1 bis 3, bei dem das Verwaltungsschnittstellenanbieter-Objekt (**26**) folgende Merkmale umfaßt:
ein Verfahren (**28**) zum Bestimmen, ob eine ME (**33**) ein zugeordnetes MI-Objekt aufweist, das einem abgefragten Merkmal entspricht;
ein Verfahren (**32**) zum Wiedergewinnen des zugeordneten MI-Objekts; und
ein Verfahren zum Auflisten von MI-Objekten, die einer ME zugeordnet sind;
wobei der Prozessor (**12**) mindestens eines der Verfahren verwendet, um das MI-Objekt, das der ME zugeordnet ist, wiederzugewinnen.

5. System gemäß einem der Ansprüche 1 bis 4, bei dem jede verwaltete Einheit (**22, 24**), die einen besonderen Dienst umfaßt, den Dienst mit einer im wesentlichen identischen ME-Schnittstelle zeigt.

6. System gemäß Anspruch 5, bei dem jede verwaltete Einheit (**22, 24**), die ein besonderes Merkmal umfaßt, das Merkmal mit einem im wesentlichen identischen MI-Objekt zeigt.

7. System gemäß Anspruch 6, bei dem der Prozessor (**12**) das Schnittstellenanbieter-Verfahren bei der Verwaltung aller Peripherieeinheiten, die ME- und MI-Objekte zeigen, einsetzt.

8. System gemäß einem der Ansprüche 1 bis 7, bei dem mindestens eine der verwalteten Einheiten (**22, 24**) mit dem

Computer (**10**) über ein Netz (**20**) verbunden ist.

9. Speichermedium (**29**) zum Steuern eines Verwaltungscomputers (**10**), um Einheiten eines Peripheriesystems zu verwalten, wobei jede verwaltete Einheit (**22, 24**) einen oder mehrere Dienste für einen Computer liefert, der der Verwaltungscomputer (**10**) sein kann oder nicht, und ferner ein Verwaltungsschnittstellenanbieter-Objekt speichert, das ermöglicht, daß Verfahren bezüglich Schnittstellenobjekten durchgeführt werden, wobei jeder Dienst eine zugeordnete Verwaltete-Entität-Schnittstelle (ME-Schnittstelle) aufweist, die einen oder mehrere Bezüge auf eine oder mehrere Verwaltungsschnittstellen (MIs) umfaßt, wobei jede MI ein oder mehrere Verfahren aufweist, um zumindest (i) Informationen bezüglich eines oder mehrerer Merkmale des Dienstes, der durch die ME (**33**) dargestellt wird und durch eine verwaltete Einheit (**22, 24**) durchgeführt wird, zu liefern, oder (ii) zu bewirken, daß die verwaltete Einheit (**22, 24**) eine Funktion durchführt, wobei der Verwaltungscomputer (**14**) eine Eingabe/Ausgabe-Einrichtung (**14**) zum Ermöglichen von Kommunikationen mit jeder der verwalteten Einheiten (**22, 24**) und einen Speicher (**16**) zum Speichern eines Explorer-Objekts (**34**) aufweist, wobei das Speichermedium (**29**) folgende Merkmale aufweist:

a) eine Einrichtung (**29**) zum Steuern des Verwaltungscomputers (**10**), um auf eine Benutzerabfrage bezüglich eines Verwaltete-Einheit-Diensts anzusprechen, um das Explorer-Objekt (**34**) aufzurufen, um eine ME (**33**), die dem Verwaltete-Einheit-Dienst entspricht, zu bestimmen; und

b) eine Einrichtung (**29**) zum Steuern der verwalteten Einheit (**22, 24**), um ein Verfahren des Schnittstellenanbieter-Objekts (**26**) aufzurufen, um ein MI-Objekt wiederzugewinnen, das der ME (**33**) zugeordnet ist, die einen Zugriff auf Informationen, die sich auf die verwaltete Einheit (**22, 24**) beziehen, ermöglicht oder bewirkt, daß die verwaltete Einheit (**22, 24**) fallabhängig eine Funktion durchführt, wobei sich die Informationen oder die Funktion auf die Benutzerabfrage beziehen.

10. Speichermedium (**29**) gemäß Anspruch 9, bei dem der Speicher (**16**) ferner ein Ansicht-Objekt (**36**) umfaßt und die Einrichtung b) den Verwaltungscomputer (**10**) steuert, um ein Verfahren aufzurufen, das dem Ansicht-Objekt (**36**) zugeordnet ist, um MEs anzuzeigen, wobei mindestens eine der MEs dem Verwaltete-Einheit-Dienst entspricht, wodurch ermöglicht wird, daß ein Benutzer eine oder mehrere der MEs vor dem Aufrufen einer Funktion des Schnittstellenanbieter-Objekts auswählt.

11. Speichermedium (**29**) gemäß Anspruch 9 oder 10, bei dem der Speicher (**16**) ferner ein Ansicht-Objekt (**36**) umfaßt und die Einrichtung b) den Verwaltungsprozessor (**10**) steuert, um ein Verfahren, das dem Ansicht-Objekt (**36**) zugeordnet ist, aufzurufen, um einen Zustand einer Gruppe von MEs anzuzeigen.

12. Speichermedium (**29**) gemäß einem der Ansprüche 9 bis 11, bei dem das Schnittstellenanbieter-Objekt (**26**) folgende Merkmale umfaßt:

ein Verfahren (**28**) zum Bestimmen, ob eine ME (**33**) ein zugeordnetes MI-Objekt aufweist, das einem abgefragten Merkmal entspricht;

ein Verfahren (**32**) zum Wiedergewinnen des zugeordneten MI-Objekts; und

ein Verfahren (**30**) zum Auflisten von MI-Objekten, die einer ME (**33**) zugeordnet sind;

wobei die Einrichtung b) die verwaltete Einheit (**22, 24**) steuert, um mindestens eines der Verfahren zu verwenden, um das MI-Objekt, das der ME (**33**) zugeordnet ist, wiederzugewinnen.

13. Speichermedium (**29**) gemäß einem der Ansprüche 9 bis 12, bei dem jede verwaltete Einheit (**22, 24**), die einen besonderen Dienst umfaßt, den Dienst mit einer im wesentlichen identischen ME-Datenstruktur zeigt.

14. Speichermedium (**29**) gemäß Anspruch 13, bei dem jede verwaltete Einheit (**22, 24**), die ein besonderes Merkmal umfaßt, das Merkmal mit einem im wesentlichen identischen MI-Objekt und enthaltenen Verfahren zeigt.

15. Speichermedium (**29**) gemäß Anspruch 14, bei dem der Prozessor das Schnittstellenanbieter-Verfahren bei der Verwaltung aller Peripherieeinheiten, die ME- und MI-Objekte zeigen, einsetzt.

Hierzu 6 Seite(n) Zeichnungen

- Leerseite -

Nummer: DE 100 06 416 A1
Int. Cl.[7]: G 06 F 13/10
Offenlegungstag: 2. November 2000

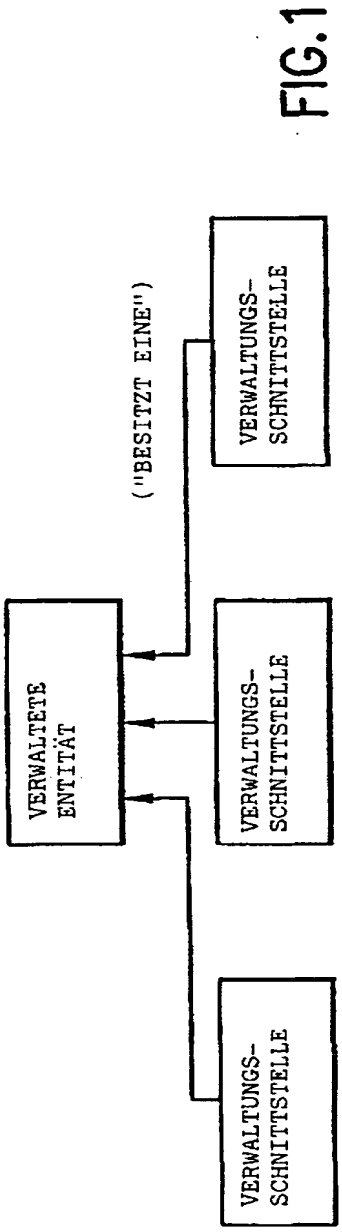

FIG.1

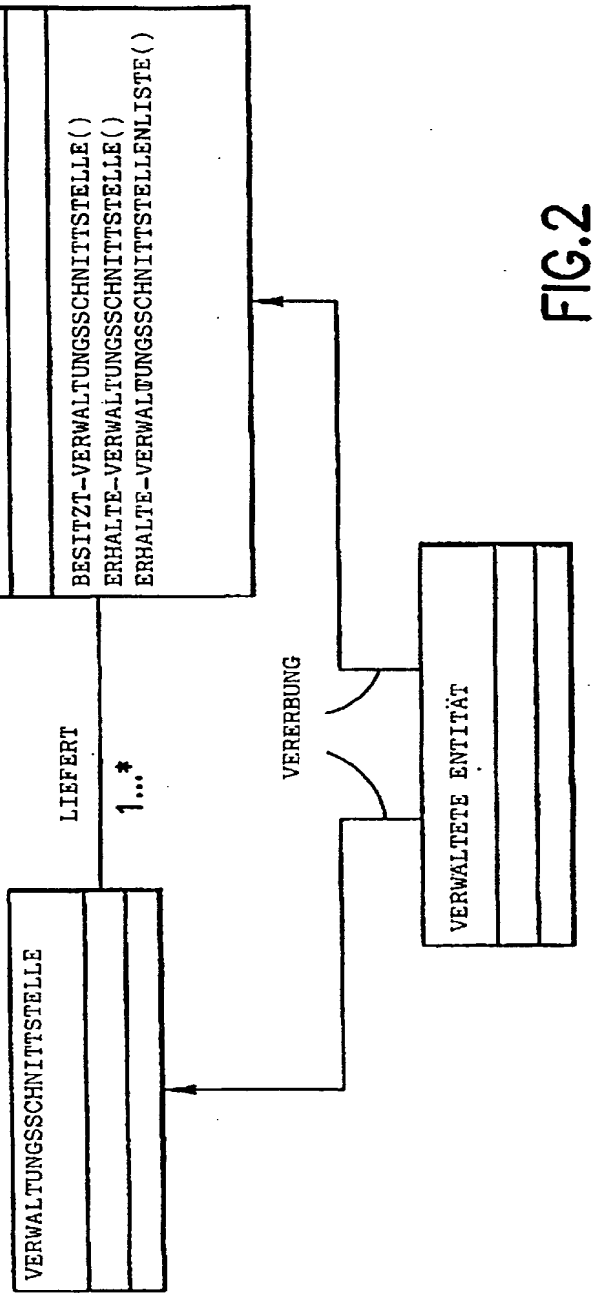

FIG.2

Nummer: DE 100 06 416 A1
Int. Cl.[7]: G 06 F 13/10
Offenlegungstag: 2. November 2000

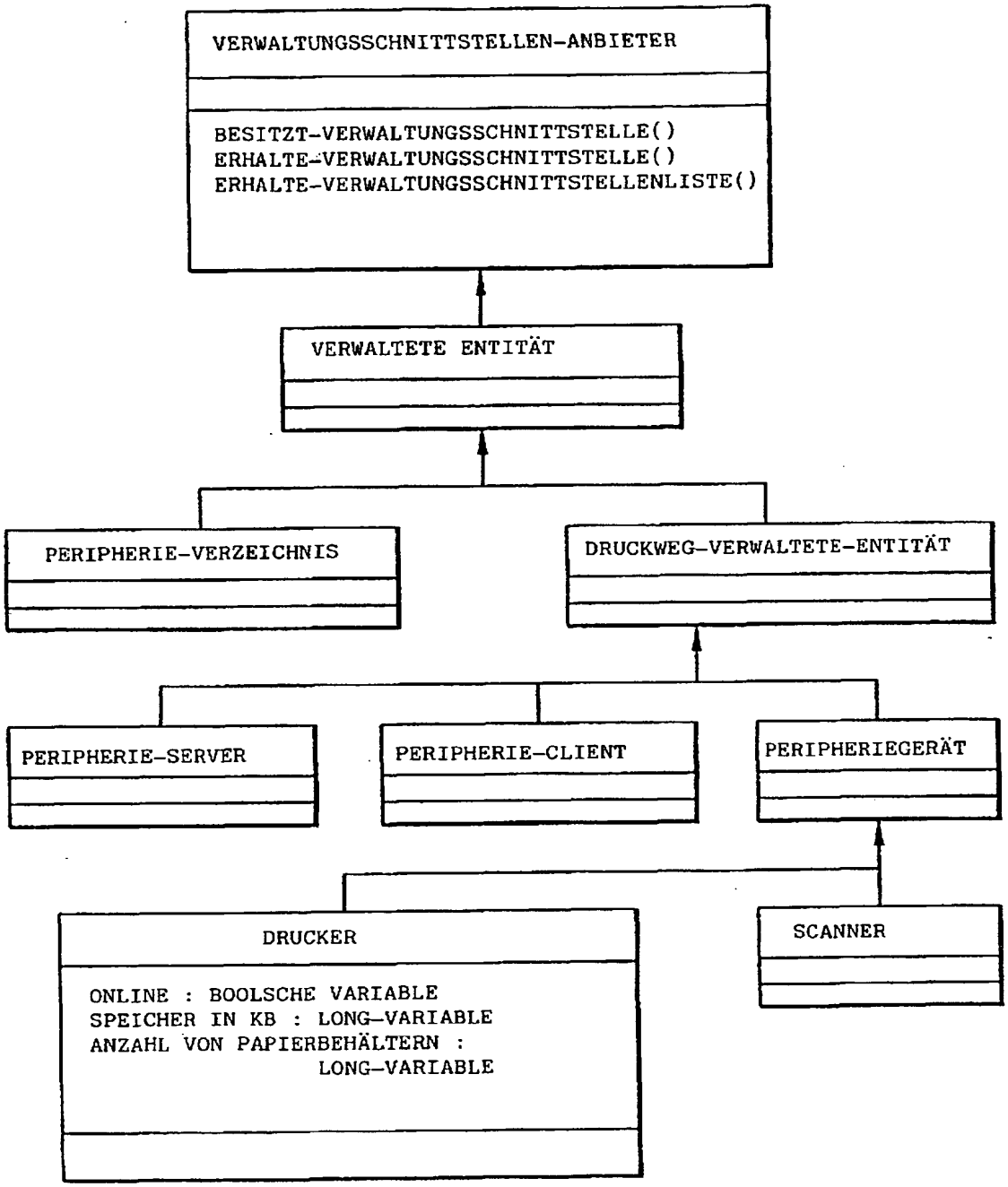

FIG.3

Nummer: DE 100 06 416 A1
Int. Cl.[7]: G 06 F 13/10
Offenlegungstag: 2. November 2000

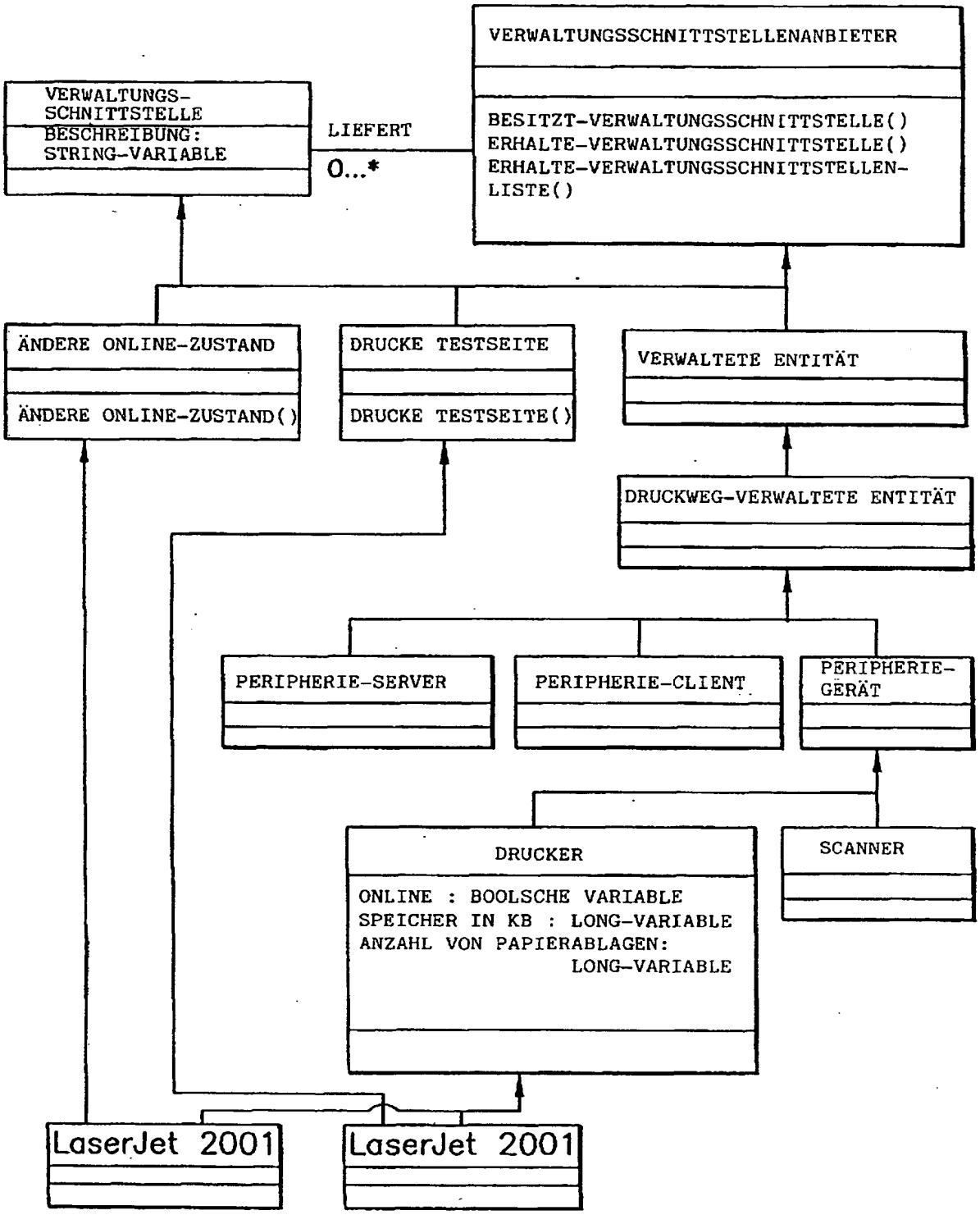

FIG.4

Nummer: DE 100 06 416 A1
Int. Cl.[7]: G 06 F 13/10
Offenlegungstag: 2. November 2000

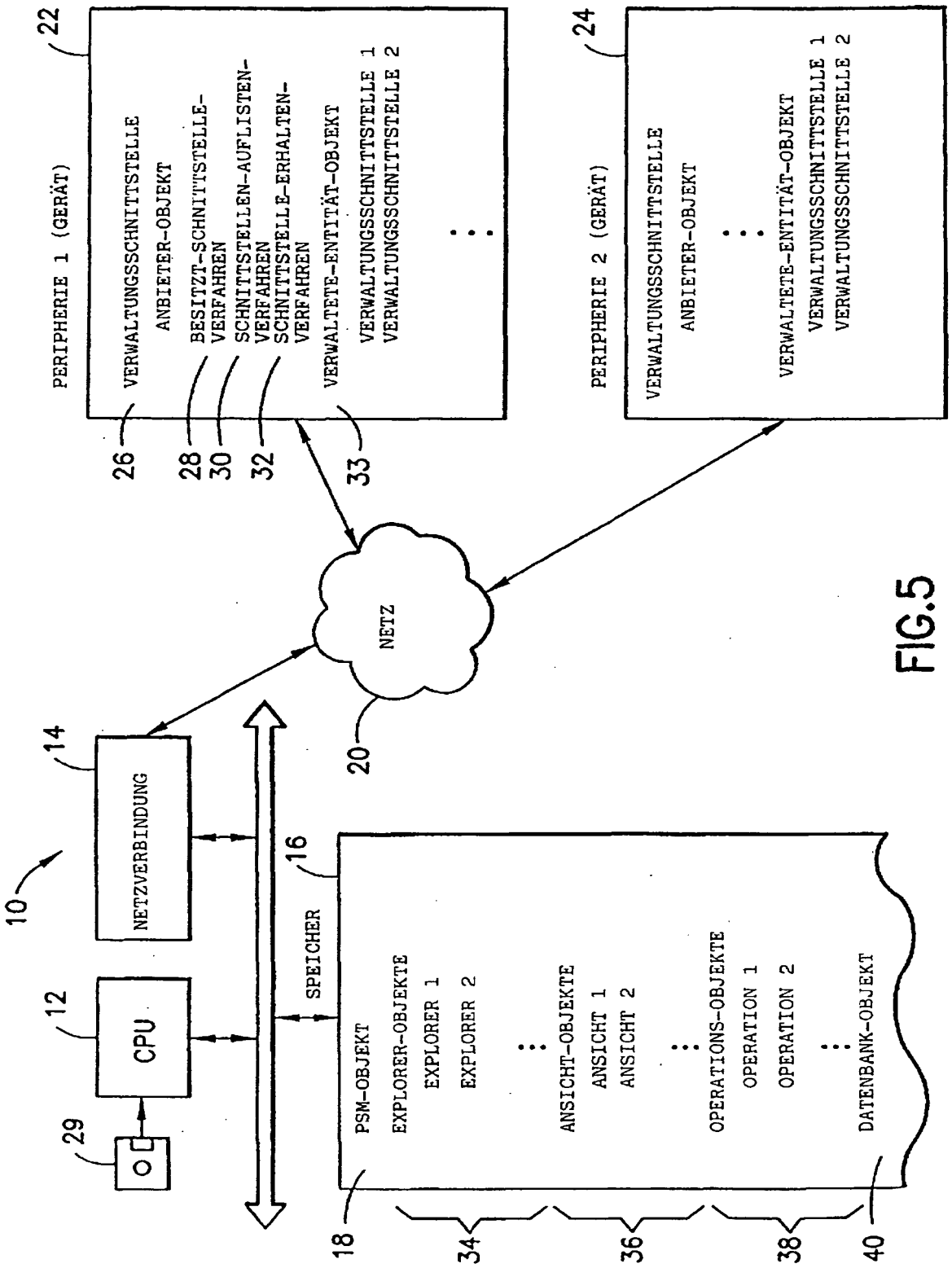

FIG.5

Nummer: DE 100 06 416 A1
Int. Cl.[7]: G 06 F 13/10
Offenlegungstag: 2. November 2000

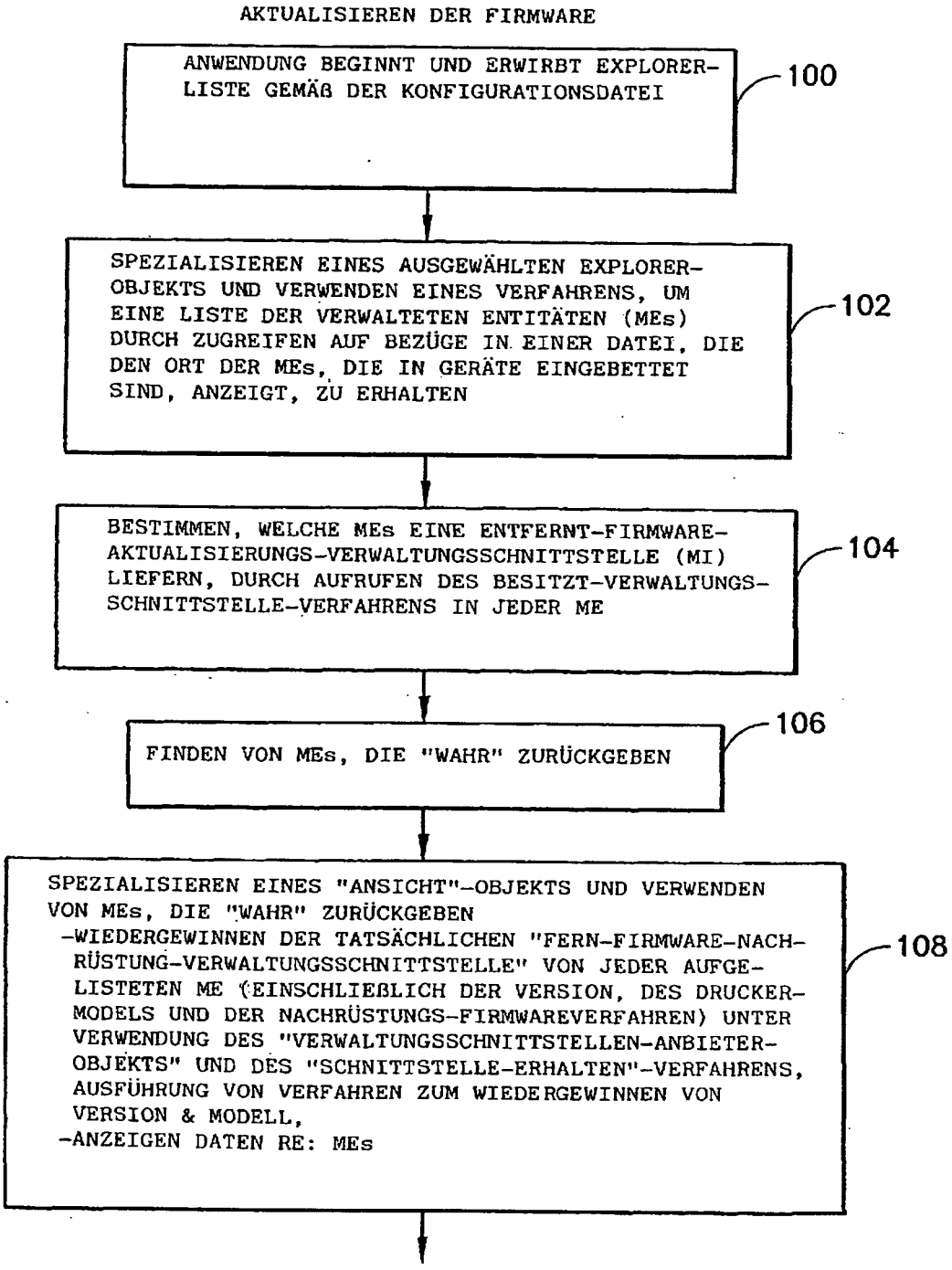

FIG.6

Nummer: DE 100 06 416 A1
Int. Cl.[7]: G 06 F 13/10
Offenlegungstag: 2. November 2000

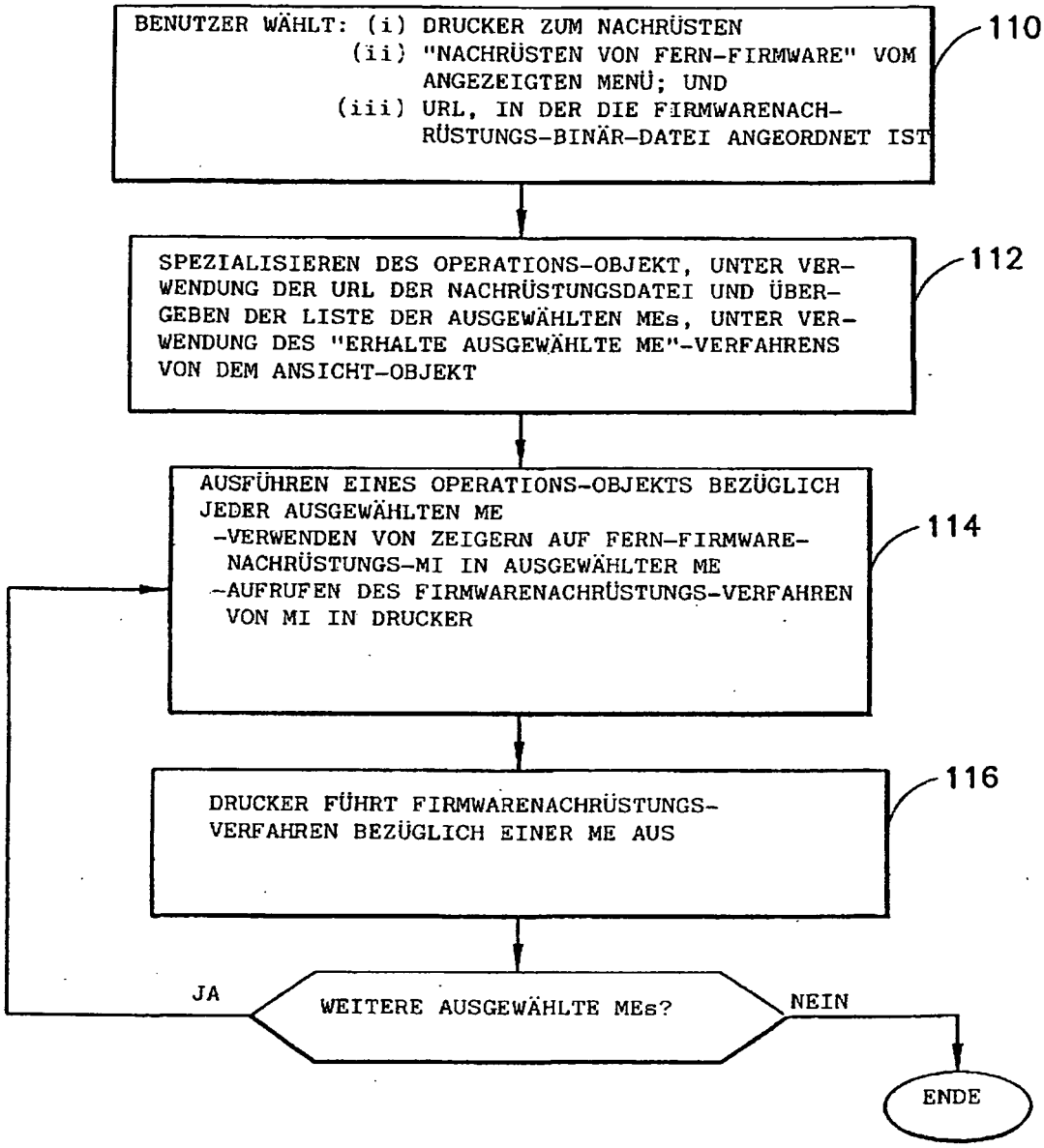

FIG.7